Qualis: the Quality of Service Component for the Globus Metacomputing System

Craig A. Lee
James Stepanek
B. Scott Michel

Computer Science and Technology
The Aerospace Corporation
El Segundo, CA 90245-4691
{lee, stepanek, scottm}@aero.org

Ian Foster
Mathematical and Computer Sciences
Argonne National Laboratory
Argonne, IL 60439
foster@ms.anl.gov

Carl Kesselman
Robert Lindell
Soonwook Hwang
Joseph Bernstein

Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292-6005
{carl, lindell, hwang, joseph}@isi.edu

Alain Roy
Department of Computer Science
University of Chicago
Chicago, IL 60637
alain@cs.uchicago.edu

1 Introduction

General computing over a widely distributed set of heterogeneous machines—typically called metacomputing—offers definite advantages. In addition to allowing a single application to bring together different types of resources, such as specialized data sources, data bases, and visualization systems, it allows an application to acquire and utilize many different machines to attain a level of compute power that is not possible any other way. It may also be more cost-effective to aggregate several machines over a network on a per-need basis rather than acquire and maintain one large machine.

The performance of metacomputing systems, however, can be highly dependent on the available network bandwidth and latencies. While performance can be improved by designing applications to adapt to the available bandwidth and tolerate latencies, the notion of quality of service (QoS) for metacomputing is important. However, QoS in a metacomputing environment has a broader scope than in just a networking environment.

Much work has been done on how to provide QoS in networks. The literature has investigated both connectionless [1] and connection-oriented [8] networks, resulting in an extremely large archive of results. Less thoroughly investigated, however, are the problems that arise in providing end-to-end QoS to large applications. The exception here, of course, is network-based multimedia, such as the popular Allophone tools [10], in which it is paramount to assure that the required QoS can be provided from a source—typically a streaming video or audio source—to one or more destinations. The service model for these tools tends to be isochronous bit rate with minimal delay jitter and low packet loss. We anticipate that metacomputing will need a substantially different service model for the non-visualization applications.

QoS specification and management can be implemented at different levels of abstraction. The Integrated Services Architecture, which includes the RSVP [2] protocol, serves as Qualis’ basic mechanism for QoS signaling protocol and traffic management. This is an example of low-level QoS mechanism that allows reservations to be made on a network for a flow between a sender and a receiver. Quality of Service for CORBA objects (QoO) [11] is an example of high-level QoS that augments the CORBA Interface Definition Language with a QoS Definition Language (QDL) that allows specification of QoS in terms of object behavior, e.g., method invocations per second.

Very large distributed computing enterprises, it has been recognized that end-to-end QoS requires that QoS be integrated within a general resource management framework. This is the goal of the Global Resource Management project [4] which plans to support command, control, communications, and intelligence (C 3I) applications, in addition to multimedia applications. The ERIS project (End-to-End Resource Management for Distributed Systems) [3] is developing an infrastructure to map end-to-end, application-level QoS specifications to network-aware, QoS, and network-level QoS specifications; allocate and schedule computing, communication and storage resources to applications; and appropriately handle QoS violations.

This position paper presents Qualis, the quality of
service component for the Globus Metacomputing system[6]. We present the Qualis architecture, how it is integrated into the Globus architecture, and how it addresses QoS in a metacomputing environment.

2 The Qualis Architecture

A guiding principle in the implementation of QoS for Globus was not to build a single, unique QoS tool, but to build an infrastructure where lower-level QoS mechanisms can be integrated and debugged, thereby allowing higher-level QoS tools and functionality to be built and evaluated. Since Nexus [7] is the communication and process control workhorse of Globus, it is clear that implementing QoS for Globus meant implementing QoS for Nexus. Hence, we chose to implement QoS for relevant Nexus abstractions. The abstractions we chose are (1) processes, (2) threads, (3) memory, and (4) communication startpoints and endpoints which are used for asynchronous Remote Service Requests (a RSRs). These are the QoS-able objects of Qualis. While Nexus does not actually have a separate abstraction for memory, it is not inconsistent with the Nexus model and will be useful since it will allow coordination of buffer management along with other forms of QoS services.

For Nexus, a process involves an address space on some processor that could be a uniprocessor, a shared-memory machine, or one node in a distributed-memory machine. A thread lives within an address space and synchronize in the usual manner with mutexes and condition variables. Communication via an RSR is done between a startpoint that is bound to an endpoint. A context may have an arbitrary number of start/endpoints. Startpoints (but not endpoints) can be freely passed among processes. One or more threads or non-threaded handlers are associated with an endpoint. An RSR is initiated with a startpoint, handler id, and a data buffer. The data buffer is then delivered to the context with bound endpoint and passed to the handler. RSRs are supported by a variety of protocol modules that utilize, for example, TCP (Transmission Control Protocol), UDP (User Datagram Protocol), MPL (the IBM Message Passing Layer), NX (the Intel Message Passing library), or shared-memory. A single context can use multiple modules depending on which one is best for the target endpoint. This approach can be used to support traditional message passing, synchronous remote procedure call, distributed shared memory, streams, multicast, and other communication and control models.

The Qualis architecture is shown in Figure 1. Nexus registers a process, thread, or bound startpoint/endpoint with QoS Lib whenever they are created as part of normal operation. They are deregistered when they are destroyed. Nexus applications can specify the QoS and call back handler associated with QoS-able objects. When this association is made, Qeli b passes the information to QoS Man which does local admission control and monitoring. If the relevant QoS mechanism requires root privilege, QoS Man will send an RPC to QoS Priv which does the privileged operation. (Modularizing privileged operations and using RPC (rather than Nexus remote service requests) was done so that only a relatively small amount of code using more familiar (and perhaps more trusted) communication mechanism would be needed, thereby mitigating any need for verification by system administrators.) If QoS Man detects a QoS violation, an ISR is sent to the application's call back handler.

Qualis is currently integrated with two low level QoS mechanisms: the POSIX Real-time Extensions [9] and RSVP [2]. The POSIX Real-time Extensions are used to control the priority of processes and threads and to lock down pages of memory. Currently, process and thread QoS is simply specified as a priority by the local QoS Man. Admission control for processes can be done by comparing the current load with the anticipated load of another process with the requested priority. Admission control for threads is done in a similar manner but can be complicated by the QoS's thread scheduling model.

QoS for startpoints/endpoints is currently based only on RSP. While RSVP supports both TCP and UDP, including multicast, only the Nexus TCP pro-

[Image Figure 1: The Qualis Architecture]
toool module has been integrated into Qualis at this
time. After the startpoint has been bound to the end-
point, the application has specified the RSVP Rs p e c
do a s T s p e c c, and the socket connection has been made,
then the RSVP signaling protocol is initiated. If the
bandwidth is available, RSVP returns a reservation
confirmation. The Alternative Queuing (AIQ) Class-
Based Queuing package [5] polices traffic and provides
static admission control. We currently have no mea-
surement based admission control (MBAC) that would
provide better utilization of the link.

We have built a wide area testbed over CARN (Co-
laborative Advanced Interagency Research Network)
using Integrated Services capable routers and hosts.
The routers were constructed using LMPC compat-
ible machines running FreeBSD 2.2.1x with a quad fast
eternet network adapter and the AIQ package which
utilizes the packet scheduler developed at LBNL, UC,
and Sun Microsystems. Our end hosts are Sun Ultra 1
workstations running the Sun Integrated Services
package. A network of two routers and three end hosts are
located at both ISI and ANL and connected by the
CARN research network.

This basic infrastructure allows a number of issues
to be investigated. QoS mapping from higher level spe-
cifications that are related to application behavior, such
as method invocations per second, to lower level sys-
tem QoS primitives is important but difficult. Any QoS
mechanism aids some overhead and, hence, implies
some minimum granularity that can realize a net gain
in performance. For threaded RSVP handlers, it may be
necessary to specify QoS prior to creation rather than
after. More effective monitoring and policing mecha-
nism need to be in place such that applications can
adapt when necessary in a timely fashion. True end-to-
end, application level QoS will require taking network,
thread, process and application behavior into account
together as a whole. Better overall performance may be
possible if multiple QoSs negotiate among themselves to
maintaining global or distributed QoS properties.

Using the wide area testbed described above, the
Globus/Qualis projects are currently performing a
number of experiments including basic RSVP
microbenchmarks and instrumented applications to eval-
uate the performance and overhead of these QoS mecha-
nisms. We are also investigating the design of QoS-
aware resource brokers and schedulers that interact
with the Ma-accomputing Directory Service and other
Globus services.

References

services in the internet architecture: An overview.

and S. Jamin. Resource reservation proto-
col (RSVP) – version 1 functional specification.
Technical report, USC Information Sciences In-
stitute, September 1997. Proposed Standard,

end resource management for distributed systems.
dex.html.

management for complex distributed systems. In
Second International Workshop on Ob-

[5] S. Floyd and V. Jacobson. Link-sharing and
resource management models for packet net-
works. IEEE/ACM Transactions on Network-

computing infrastructure toolkit. Intl. J. Superc-

approach to integrating multithreading and com-
nunication. J. Parallel and Distribut-
eing, 37:70-82, 1996.

[8] M. Garrett. ALM service architecture: Fromap-
plications to scheduling, 1994. ALMForumCon-
tribution 94-0846 TMSW.

[9] IEEE. Pt003.4 Real-time extensions for portable
Standard, http://standards.ieee.org/cata-
log/posix.html.

work for for packet video. In Proc. ACM Multi-
time media '95, 1995.

chitectural support of quality of service. Theory
and Practice of Object Systems, 3(1)